Using the
PABO
Digital Learning Computer

Welcome to the world of the PABO Digital Learning Computer, a device designed to teach the amateur the basics of programming a modern digital computer system.

This guide is intended for persons already familiar with the operation of digital computers in general. For persons requiring a more thorough grounding in the underlying concepts involved, we suggest starting out with PABO Manual 1: *Introduction to Digital Computers* (MPN: PDC-M01).

Architecture:
The PABO Digital Learning Computer is a simple 12-bit Von Neuman style computer with a Turing Complete instruction set featuring 8 operations. The PABO's 12-bit word is divided between a 3-bit opcode and an 9-bit memory address affording an address space of 512 words.

Registers:
All operations are performed using a 9-bit Program Counter (PC), a 9-bit Accumulator (A), and a Zero Bit (Z) that can test the contents of the Accumulator.

Instruction Formats:

Bits [11..9]	[8..0]	[opcode] [address]	"General"
Bits [11..9]	[8..0]	[000] [microcode]	"OPeRate"

Command Set:
The PABO can execute instructions to Load or Store data, Add or Subtract values, and Jump conditionally. There are also commands to Test the value in the Accumulator, logically Not the bits of the Accumulator, and Halt processing.

Notation:
Here are the essentials of the ADD instruction. (ADD 0o4xxx A = A + [xxx]) “ADD” represents the English name for the instruction. “0o4xxx” denotes the machine code format of the instruction. An 0o4 in the highest order 3-bits of the memory location indicates an ADD instruction. “A=A+[xxx]” describes the ADD operation. In this case the lowest order 9-bits of the instruction word are a memory address. The contents of that address are ADDed to the value in the Accumulator.

The PABO Command Set:

LDA	0o2xxx	A = [xxx]
STO	0o5xxx	[xxx] = A
ADD	0o4xxx	A = A + [xxx]
SUB	0o6xxx	A = A - [xxx]
JNZ	0o1xxx	If Z==1, PC = xxx
TST	0o0001	If A==0, Z=1 else Z=0
NOT	0o0002	A = !A
HLT	0o0777	Halt

Sample Programs:

```
# PROBLEM:
# For any number (n), compute the sum of all integers 1..n
# :BEGIN
memory[0o0]= 0o2011   # LDA NUMBER
memory[0o1]= 0o4012   # ADD SUM
memory[0o2]= 0o5012   # STO SUM
memory[0o3]= 0o2011   # LDA NUMBER
memory[0o4]= 0o6013   # SUB ONE
memory[0o5]= 0o5011   # STO NUMBER
memory[0o6]= 0o0001   # TST
memory[0o7]= 0o1000   # JNZ BEGIN
memory[0o10]= 0o0777 # HLT
# :DATA
memory[0o11]= 0o06    # 'NUMBER
memory[0o12]= 0o00    # 'SUM
memory[0o13]= 0o01    # 'ONE
```

```
# PROBLEM: Push three values to a stack
# MAIN:
# PUSH WORD TO STACK
memory[0o000] = 0o2000     # LDA [this word]
memory[0o001] = 0o5016     # STO [top of stack]
# UPDATE STACK POINTER
memory[0o002] = 0o2001     # LDA [stack pointer]
memory[0o003] = 0o6010     # SUB [1 from stack pointer]
memory[0o004] = 0o5001     # STO [stack pointer]
# UPDATE LOOP COUNTER
memory[0o005] = 0o2017     # LDA [loop counter]
memory[0o006] = 0o6010     # SUB [1 from loop counter]
memory[0o007] = 0o5017     # STO [loop counter]
memory[0o010] = 0o0001     # TST [is loop counter 0?]
memory[0o011] = 0o1000     # JNZ [if not push another value]
memory[0o012] = 0o0777     # HLT

memory[0o017] = 0o0003     # LOOP COUNTER
```

So here’s a funny thing. The PABO Digital Learning Computer actually isn’t a physical device but a modern and painfully simple Python program. Here it is.

```
#
#  The PABO Digital Learning Computer
#
import time             # FOR DEBUG SLEEP
import sys              # FOR SYS.EXIT

memory = [0] *512   # MEMORY
a = 0o0                 # ACCUMULATOR
pc = 0o0                # PROGRAM COUNTER
z=0                     # ZERO FLAG
o=0                     # OVERFLOW FLAG (UNUSED ATM)

# HERE IS THE CONTENTS OF memory
# YOU ‘PROGRAM’ PABO WITH THESE ASSIGNMENTS
# MAIN:
memory[0o0000] = 0o2017 # LDA #3
memory[0o0001] = 0o0002 # NOT A
memory[0o0002] = 0o4015 # ADD #1
memory[0o0003] = 0o4016 # ADD #2
memory[0o0004] = 0o0777 # HLT
# DATA
memory[0o0015] = 0o0001
memory[0o0016] = 0o0002
memory[0o0017] = 0o0013


#
# MAIN
#

while True:

        op = memory[pc]>>9

# HLT  (OPR 0o0777)
        if (memory[pc] == 0o0777):
                break
# TST  (OPR 0o0001)
        elif (memory[pc] == 0o0001):
                if (a == 0):
                        z=1     # a == 0
                else:
                        z=0     # a != 0
                pc = pc + 1
# NOT (OPR 0o0002)
        elif (memory[pc] == 0o0002):
                a = 0o7777 - a
                pc = pc + 1

# JNZ
        elif (op == 1):                 # JUMP IF TST (a != 0) RETURNED TRUE
                if (z == 1):    # Z flag set (a == 0)
                        pc = pc + 1
                else:                   # Z flag clear (a != 0) *JUMP*
                        pc = memory[pc] & 0o0777
```

```
# LDA
        elif (op == 2):                 # LOAD ACCUMULATOR
                target = memory[pc] & 0o0777
                a = memory[target]
                pc = pc + 1
# ADD
        elif (op == 4):                 # ADD ACCUMULATOR
                target = memory[pc] & 0o0777
                a = a + memory[target]
                if (a > 0o7777):
                        a = a - 0o7777
                        o = 1
                pc = pc + 1
# STA
        elif (op == 5):                 # STORE ACCUMULATOR
                target = memory[pc] & 0o0777
                memory[target] = a
                pc = pc + 1
# SUB
        elif (op == 6):                 # SUB ACCUMULATOR
                target = memory[pc] & 0o0777
                a = a - memory[target]
                pc = pc + 1

sys.exit()
```